Information-Based Compact Pose SLAM

Viorela Ila, Josep M. Porta, and Juan Andrade-Cetto, *Member, IEEE*

***Abstract*—Pose SLAM is the variant of simultaneous localization and map building (SLAM) is the variant of SLAM, in which only the robot trajectory is estimated and where landmarks are only used to produce relative constraints between robot poses. To reduce the computational cost of the information filter form of Pose SLAM and, at the same time, to delay inconsistency as much as possible, we introduce an approach that takes into account only highly informative loop-closure links and nonredundant poses. This approach includes constant time procedures to compute the distance between poses, the expected information gain for each potential link, and the exact marginal covariances while moving in open loop, as well as a procedure to recover the state after a loop closure that, in practical situations, scales linearly in terms of both time and memory. Using these procedures, the robot operates most of the time in open loop, and the cost of the loop closure is amortized over long trajectories. This way, the computational bottleneck shifts to data association, which is the search over the set of previously visited poses to determine good candidates for sensor registration. To speed up data association, we introduce a method to search for neighboring poses whose complexity ranges from logarithmic in the usual case to linear in degenerate situations. The method is based on organizing the pose information in a balanced tree whose internal levels are defined using interval arithmetic. The proposed Pose-SLAM approach is validated through simulations, real mapping sessions, and experiments using standard SLAM data sets.**

***Index Terms*—Information filter, information gain, interval arithmetic, Pose SLAM, state recovery, tree-based data association.**

I. Introduction

THE USE of information-based representations is one of the keys to obtaining a computationally efficient on-line solution to simultaneous localization and map building (SLAM) [1]–[4]. When estimating both the last robot pose and a map of features, the ensuing information matrix turns out to be approximately sparse with very small matrix entries for distant landmarks [1]. Exactly sparse information matrices can be obtained by estimating the entire robot path along with the map, which is an approach typically referred to as full SLAM [4]–[6]. Exact sparsity can also be achieved using a small set of active landmarks to relocate the robot from scratch at every iteration [7], by decoupling the estimation problem maintaining the map only [8] or, as in Pose SLAM, by maintaining only the robot trajectory [2], [3], [9]. This last variant of SLAM relies on the idea that landmark locations can be easily retrieved once the robot path has been properly estimated [10]. Thus, in Pose SLAM, landmarks are used only to derive relative measurements linking pairs of poses.

When working with sensors that are able to identify many landmarks per pose, Pose SLAM produces more compact maps than the other exactly sparse approaches, resulting in better performance. However, Pose SLAM also presents some drawbacks: Adding all the robot poses to the map has the cost of a representation that grows independently of the size of the area to map, and adding all possible links reduces the sparsity of the information matrix, slowing down the execution. Furthermore, when using a linearized approach, the accumulation of linearization errors introduced by each new link produces overconfident estimates, which lead to filter inconsistency [11], [12].

In this paper, we introduce a principled on-line approach for Pose SLAM, which only keeps nonredundant poses and highly informative links (see Fig. 1). This is achieved by computing two measures: the distance between a given pair of poses and the mutual information gain when linking two poses. In this paper, we show that, in Pose SLAM, the exact form of these two measures can be computed in constant time. When compared with the existing approaches [2], the proposed system produces a more compact map that translates into a significant reduction of the computational cost and a delay of the filter inconsistency, maintaining the quality of the estimation for longer mapping sequences.

By applying the proposed strategy, the robot closes only few loops and it operates in open loop for long periods, which is feasible using recent odometric techniques [13], [14]. The proposed Pose-SLAM method, which is presented in this paper, includes a novel-state-recovery procedure at loop closure that takes advantage of the inherent sparsity of the information matrix scaling linearly both in time and memory. This computational cost is further amortized over the period where the robot operates in open loop, for which we introduce a factorization of the cross covariance that allows the state to be updated in constant time. Thus, the proposed state-recovery strategy outperforms state-of-the-art approaches that take linear time for very sparse matrices (i.e., when the robot operates in open loop) but are worst-case quadratic when many loops are closed [2], [15].

Manuscript received April 9, 2009; revised July 29, 2009. First published November 10, 2009; current version published February 9, 2010. This paper was recommended for publication by Associate Editor C. Stachniss and Editor W. K. Chung upon evaluation of the reviewers' comments. This work was supported by Project DPI-2007-60858, Project DPI-2008-06022, Project Multimodal Interaction in Pattern Recognition and Computer Vision Consolider-Ingenio 2010, and by the European Union Ubiquitous Networking Robotics in Urban Settings under Project IST-FP6-STREP-045062. The work of V. Ila was supported in part by the Spanish Ministry of Science and Innovation under a Juan de la Cierva Postdoctoral Fellowship.

V. Ila is with the Institut de Robòtica i Informàtica Industrial, Spanish National Research Council, Barcelona 08028, Spain, and also with the College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: vila@iri.upc.edu).

J. M. Porta and J. Andrade-Cetto are with the Institut de Robòtica i Informàtica Industrial, Spanish National Research Council, Barcelona 08028, Spain (e-mail: porta@iri.upc.edu; cetto@iri.upc.edu).

The paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The material includes videos of the mapping sessions presented in the paper as well as the MATLAB code for the simulated experiments. This size is 22 MB. Contact vila@iri.upc.edu for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2009.2034435

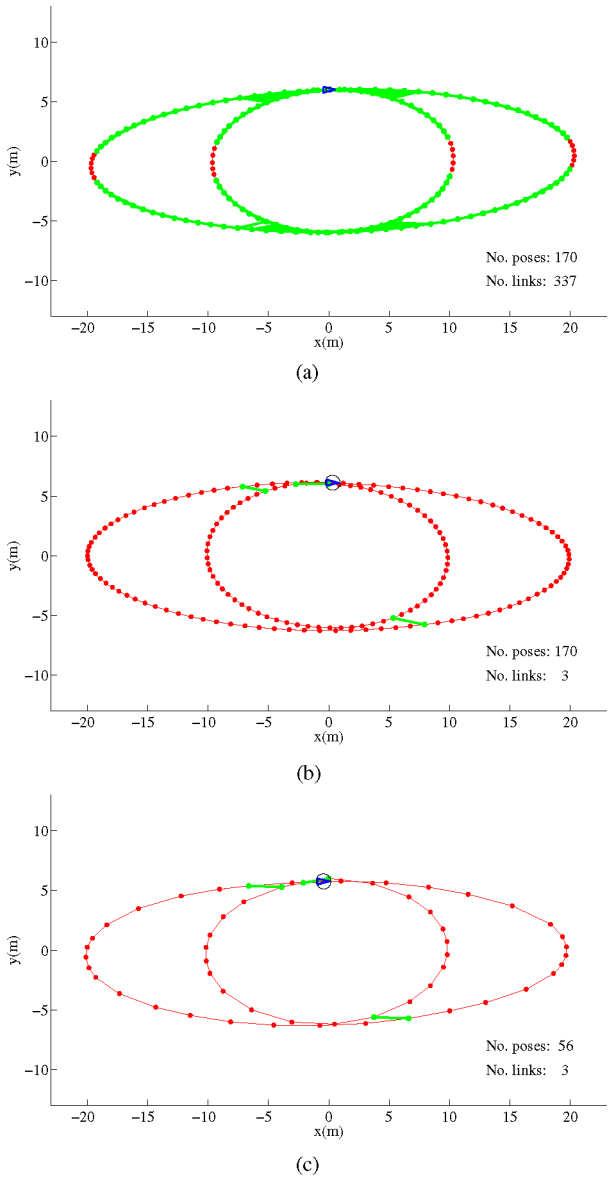

Fig. 1. Simulations that exemplify the effects of limiting the number of poses (in red) and linked poses (in green) in Pose SLAM. (a) Incorporating all poses and all links to the filter. (b) Incorporating all poses but only highly informative links. (c) Incorporating only relevant poses and highly informative links.

With the proposed technique, the bottleneck for real-time execution is not state recovery but data association, that is, detecting poses close to the current one for which feature matching is likely. Exploiting the proposed factorization of the cross covariance, we introduce a tree-based method for data association using interval arithmetic to encode the internal nodes of the tree. The main advantage of the proposed method is that it improves the search up to logarithmic time. Moreover, by taking into account the cross covariances from the very beginning, it also avoids false positives, which are typically present in existing tree-based data-association techniques [16]. This offers the possibility to use Pose SLAM to map large-scale environments.

The rest of the paper is structured as follows. In Section II, a succinct related work review is presented. Section III describes basic preliminaries and our notation for Pose SLAM. The following three sections present the main novelties of the proposed approach: the procedures to identify relevant poses and links (Section IV), the efficient state recovery (Section V), and the tree-based data association method (Section VI). Section VII summarizes the Pose-SLAM algorithm introduced in this paper and Section VIII evaluates it using synthetic and real data sets. Concluding remarks are given in Section IX.

II. Related Work

Solutions to the SLAM problem have evolved over the years, both with respect to the type of state representation, as well as with respect to the estimation tools used. Seminal solutions formulate the problem as the probabilistic estimation of the robot pose and the location of static landmarks in the environment, jointly modeled as a multivariate Gaussian, and the extended Kalman filter (EKF) became the estimation tool of choice [17], [18]. However, maintaining the fully correlated covariance matrix of the robot pose and the map of features has memory and computational complexity that scales quadratically with the number of features, limiting the approach to relatively small environments.

On the other hand, information-based representations use the inverse of the covariance matrix to sparsely encode correlations. In the case of Pose SLAM, the information matrix scales with the number of poses and it has non-null entries only for those pairs of poses directly related by a relative motion constraint. Therefore, the matrix is basically tridiagonal (consecutive poses in the trajectory are always related), and off-diagonal blocks appear when there is a link closing a loop. The advantage of a representation that grows with every new pose is that the resulting information matrix is inherently sparse. The problem is that, in a naive implementation of Pose SLAM, the state grows indefinitely, independent of the size of the area to map.

Heuristic strategies can be found in the Pose-SLAM literature to either reduce the size of the state representation by keeping only one pose every few meters [9], [19] or to restrict the number of links to a number linearly dependent on the number of poses [2]. In the context of active SLAM, principled information-based measures have been used to select actions so that the subsequent observations maximally reduce uncertainty of the map [20], [21]. In landmark-based SLAM, information-based approaches have also been proposed to reduce the state size and to delay inconsistency incorporating only highly informative observations to the filter [22], [23]. Following the same reasoning, our previous paper [3] pointed out that, in Pose SLAM, the computational complexity can be reduced, and inconsistency can be delayed, by considering only highly informative links between nearby poses. Whereas, in our previous work, we approximated the distance between poses and the information gain for links, in this paper, we propose a method to compute these two measures in exact form. The two measures require the computation of the joint marginals between the current robot pose and the previous poses along the trajectory, which is something that is not directly available in information-based representations.

Information-based approaches are perfectly suited for off-line maximum likelihood estimation. In this case, data association is usually taken for granted, joint marginals are not necessary, and the estimate only includes the state mean, which is iteratively approximated representing the relative displacement between poses as quadratic constraints [19], [24]–[26] or by factorizing the sparse information matrix [6]. On-line approaches to Pose SLAM rely either on variants of the batch methods [4] or on filtering [2], [3]. In all on-line approaches, joint marginals are needed for prior-based data association. These can be approximated in linear time using belief propagation [27], [28], in logarithmic time by subsampling poses and performing relaxation over multiple spatial resolutions [25] or in constant time by considering only first-order relations via Markov blankets [1] or by implementing partial state updates [29]. However, overconfident approximation of joint marginals may produce false negatives during data association, losing the opportunity to significantly compensate the accumulated estimation error. On the other hand, optimistic approximations of joint marginals produce false positives [30], [31] that increase the number of attempts for sensor registration, which in general is costly and subject to perceptual aliasing. Thus, exact cross covariances are preferred for accurate data association. These can be recovered by augmenting the sparse system of equations needed for state recovery [2], by exploiting the sparseness of factorized forms of the information matrix with QR [4], or by Cholesky factorizations [32]. However, these algorithms have on average linear computational complexity when moving in open loop (i.e., for band tridiagonal matrices) and are worst-case quadratic for matrices encoding many loops.

Loops are detected during data association by matching sensor readings. Data association is typically implemented as a linear scan over all previous poses, either by directly searching for feature matches in a sensor database, independent of the filter estimates [33], or by first using filter information to constrain the search for sensory matches only to few neighboring poses [2], [3]. For consistent estimates, the second option is more efficient and less affected by perceptual aliasing. With a linear time complexity to recover joint marginals, it did not make sense to implement a search for neighboring poses better than the basic linear scan. However, exploiting the cross covariance factorization presented in this paper, it now makes sense to improve the data-association process. One possibility would be to use a KD-tree [16] or a grid structure [34] to speed up nearest neighbor search. If covariances cannot be assumed bounded, tree-based techniques outperform grid-based ones both in terms of memory and execution time. In both approaches, however, poses are considered in marginal form, disregarding cross covariances and producing conservative estimates that result in false positives.

III. Pose-SLAM Preliminaries

In the on-line form of Pose SLAM [2], [3], the objective is to incrementally compute an estimate of the robot trajectory $\mathbf{x}_n = [x_0^\top, \ldots, x_n^\top]^\top$, with each x_i a random vector corresponding to the ith robot pose. The robot trajectory is updated with a set $\mathbf{z}_n$ of independent observations on the relative displacement between the current robot pose and previous poses along the path. Using a Bayesian factorization

$$p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n) \propto\ p(\mathbf{x}_n|\mathbf{x}_{n-1})\ p(\mathbf{z}_n|\mathbf{x}_n). \tag{1}$$

The set $\mathbf{z}_n$ can be split in two disjoint groups, i.e., a set of observations between the current robot pose and the immediate previous one $\mathbf{u}_n$ and a set of observations linking the current pose with any other pose, except for the previous one $\mathbf{y}_n$. The probabilistic model then becomes

$$\begin{aligned} p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n) &\propto\ p(\mathbf{x}_n|\mathbf{x}_{n-1})\ p(\mathbf{u}_n, \mathbf{y}_n|\mathbf{x}_n) \\ &\propto\ p(\mathbf{x}_n|\mathbf{x}_{n-1})\ p(\mathbf{u}_n|\mathbf{x}_n)\ p(\mathbf{y}_n|\mathbf{x}_n) \end{aligned}$$

and by using (1) for the observations on consecutive poses

$$p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{z}_n) \propto\ p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{u}_n)\ p(\mathbf{y}_n|\mathbf{x}_n).$$

The aforementioned state transition probability is factorized on the typical SLAM operations of augmenting the state $p(\mathbf{x}_n|\mathbf{x}_{n-1}, \mathbf{u}_n)$ and updating it $p(\mathbf{y}_n|\mathbf{x}_n)$.

Simultaneous observations coming from different sensors are considered independent and can be fused before using them to update the filter. As a consequence, we can assume the set $\mathbf{u}_n$ to include a single element u_n and the current pose n to be linked to a previous pose i with only a single constraint y_n^i, if available. Moreover, assuming Gaussian probability distributions $p(x_i) \sim \mathcal{N}(x_i; \mu_i, \mathbf{\Sigma}_{ii}), p(u_n) \sim \mathcal{N}(u_n; \mu_u, \mathbf{\Sigma}_u)$, and $p(y_n^i) \sim \mathcal{N}(y_n^i; \mu_y, \mathbf{\Sigma}_y)$, the state estimation can be parametrized using the information from $p(\mathbf{x}) \sim \mathcal{N}^{-1}(\mathbf{x}; \boldsymbol{\eta}, \mathbf{\Lambda})$, where the information vector $\boldsymbol{\eta}$ and the information matrix $\mathbf{\Lambda}$ are related to the mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$ by the equivalences $\boldsymbol{\mu} = \mathbf{\Sigma}\,\boldsymbol{\eta}$ and $\mathbf{\Sigma} = \mathbf{\Lambda}^{-1}$, respectively.

A. State Augmentation

The observation u_n is used to augment the state with a new pose. The state transition model is given by

$$\begin{aligned} x_n &= f(x_{n-1}, u_n) \\ &\approx f(\mu_{n-1}, \mu_u) + \mathbf{F}_n\ (\mathbf{x}_{n-1} - \boldsymbol{\mu}_{n-1}) + \mathbf{W}_n(u_n - \mu_u) \end{aligned}$$

with $\mathbf{F}_n$ and $\mathbf{W}_n$ the Jacobians of f with respect to x_{n-1} and u_n evaluated at μ_{n-1} and μ_u, respectively. With $\mathbf{Q} = \mathbf{W}_n \mathbf{\Sigma}_u \mathbf{W}_n^\top$, the state parameterized in information form is augmented as

$$\boldsymbol{\eta}_n = \begin{bmatrix} \boldsymbol{\eta}_{1:n-2} \\ \eta_{n-1} - \mathbf{F}_n{}^\top \mathbf{Q}^{-1}\ (f(\mu_{n-1}, \mu_u) - \mathbf{F}_n\ \mu_{n-1}) \\ \mathbf{Q}^{-1}\ (f(\mu_{n-1}, \mu_u) - \mathbf{F}_n\ \mu_{n-1}) \end{bmatrix} \tag{2}$$

and

$$\mathbf{\Lambda}_n = \begin{bmatrix} \mathbf{\Lambda}_{1:n-2,1:n-2} & \mathbf{\Lambda}_{1:n-2,n-1} & \mathbf{0} \\ \mathbf{\Lambda}_{n-1,1:n-2} & \mathbf{\Lambda}_{n-1\ n-1} + \mathbf{F}_n^\top \mathbf{Q}^{-1} \mathbf{F}_n & -\mathbf{F}_n^\top \mathbf{Q}^{-1} \\ \mathbf{0} & -\mathbf{Q}^{-1} \mathbf{F}_n & \mathbf{Q}^{-1} \end{bmatrix} \tag{3}$$

where η_{n-1} and $\mathbf{\Lambda}_{n-1\ n-1}$ are used to denote the blocks of $\boldsymbol{\eta}_{n-1}$ and $\mathbf{\Lambda}_{n-1}$ corresponding to the $(n-1)$th pose, and $\boldsymbol{\eta}_{1:n-2}$ and $\mathbf{\Lambda}_{1:n-2,1:n-2}$ indicate those ranging from the first to the $(n-2)$th pose.

In the basic form of Pose SLAM [2], the state is augmented every time the robot moves, even when revisiting areas. Finally, observe that only if the mean state is available, the operations in (2) and (3) can be computed in constant time.

B. State Update

Each set of measures $\mathbf{y}_n = \{y_n^i, \ldots, y_n^k\}$ constrains the relative position of the last pose with respect to some other poses from the robot trajectory, forming loops. The measurement model for each of these constraints is

$$\begin{aligned} y_n^i &= h(x_i, x_n) + \mathbf{v}_n \\ &\approx h(\mu_i, \mu_n) + \mathbf{H}(\mathbf{x}_n - \boldsymbol{\mu}_n) + \mathbf{v}_n \end{aligned}$$

where h gives the displacement from x_i to x_n in the frame of reference of x_i, and $\mathbf{H}$ is

$$\mathbf{H} = [\mathbf{0}, \ \ldots \mathbf{0} \ \mathbf{H}_i \ \mathbf{0}, \ \ldots \ \mathbf{0} \ \mathbf{H}_n] \tag{4}$$

with $\mathbf{H}_i$ and $\mathbf{H}_n$ are the Jacobians of h with respect to x_i and x_n, and $\mathbf{v}_n$ is the zero mean measurement noise with covariance $\boldsymbol{\Sigma}_y$.

The information from observation y_n^i is fed to the filter by adding the following increments:

$$\Delta\boldsymbol{\eta} = \mathbf{H}^\top \boldsymbol{\Sigma}_y^{-1} \left(y_n^i - h\left(\mu_i, \mu_n\right) + \mathbf{H}\,\boldsymbol{\mu}_n\right)$$

and

$$\Delta\boldsymbol{\Lambda} = \mathbf{H}^\top \boldsymbol{\Sigma}_y^{-1} \mathbf{H} \tag{5}$$

to $\boldsymbol{\eta}_n$ and $\boldsymbol{\Lambda}_n$, respectively.

In the basic form of Pose SLAM, the state update is applied for every loop-closure constraint obtained from sensor readings. Due to sensor limitations, only poses that are relatively close are linked, and therefore, the basic form of Pose SLAM produces an inherently sparse information matrix, even if the robot retraverses previously mapped areas. Only in degenerate situations that can be easily detected and avoided, the information matrix can get fully populated, e.g., when the robot nearly stands still and all poses in the trajectory are within the sensor range and can be linked with the current pose.

Note that each state update changes the entire state estimate. Therefore, the next filtering step requires to recover the mean (to evaluate Jacobians) and the covariance (to perform data association), which is costly in information form. In the following sections, we show how to exploit the filter information to reduce the number of updates and how to make them more efficient by reducing the size of the state.

IV. Identifying Relevant Poses and Informative Links

The strategy, we propose in order to obtain a compact and efficient Pose SLAM is based on adding only nonredundant poses and highly informative links. A new pose is considered redundant when it is too close to another pose already in the trajectory, and not much information is gained by linking this new pose to the map. However, if the new pose allows the establishment of an informative link, both the link and the pose are added to the map. The result is a uniform distribution of poses in the information space and not in the Euclidean space, as proposed by existing approaches [9], [19].

Next, we describe how to compute the distance between poses and the information gain when a link is to be established.

A. Distance Between Two Poses

The relative displacement d from the current robot pose x_n to any other previous pose in the trajectory x_i can be estimated as a Gaussian with parameters

$$\begin{aligned} \mu_d &= h(\mu_i, \mu_n) \\ \boldsymbol{\Sigma}_d &= [\mathbf{H}_i \quad \mathbf{H}_n] \begin{bmatrix} \boldsymbol{\Sigma}_{ii} & \boldsymbol{\Sigma}_{in} \\ \boldsymbol{\Sigma}_{in}^\top & \boldsymbol{\Sigma}_{nn} \end{bmatrix} [\mathbf{H}_i \quad \mathbf{H}_n]^\top \end{aligned}$$

where $\boldsymbol{\Sigma}_{in}$ is the cross correlation between the ith and the current pose.

We marginalize the distribution on the displacement for each one of its dimensions r to get a 1-D Gaussian distribution $\mathcal{N}(\mu_r, \sigma_r^2)$ that allows to compute the probability of pose x_i being closer than v_r to pose x_n along dimension r

$$\begin{aligned} p_r &= \int_{-v_r}^{+v_r} \mathcal{N}\left(\mu_r, \sigma_r^2\right) \\ &= \frac{1}{2}\left(\operatorname{erf}\left(\frac{v_r - \mu_r}{\sigma_r\sqrt{2}}\right) - \operatorname{erf}\left(\frac{-v_r - \mu_r}{\sigma_r\sqrt{2}}\right)\right). \end{aligned} \tag{6}$$

If, for all dimensions, p_r is above a given threshold s, then pose x_i is considered close enough to the current robot pose x_n. If no pose is close to x_n, it is included in the map using the procedure described in Section III-A.

Thresholds v_r are defined from the sensor characteristics (field of view for cameras, maximum distance considered in the laser alignment, etc.) In general, it is experimentally simpler to define a threshold for each dimension separately than to define a single threshold for a measure integrating the distances along all dimensions (e.g., a weighted norm). Thus, even if there is no technical difficulty to integrate the distance measures for the different dimensions, it is more practical to consider them separately. This way, we also avoid mixing rotational and translational dimensions.

With respect to threshold s, it would be desirable to adjust it with the uncertainty in the robot pose, decreasing it as the robot gets lost. Nevertheless, to avoid introducing more complexity, we set this constant low in our experiments, i.e., 0.1. With such a conservative value for s, only poses that are for sure far away from the current pose do not pass the distance test, and no possible loop closures are missed.

B. Mutual Information Gain for Candidate Links

The mutual information gain for a candidate link measures the amount of uncertainty removed from the state when the link is integrated into the filter. For Gaussian distributions, it is given by the logarithm of the ratio of determinants of prior and posterior state covariances [20]–[23]. These determinants are proportional to the volume of the covariance hyper-ellipsoids of equiprobability. Thus, this ratio is related with the number of times the state uncertainty shrinks once a loop is asserted.

As the covariance matrix is the inverse of the information matrix and taking into account (5), we have that the mutual information gain of a candidate link between poses i and n is

$$\mathcal{I} = \frac{1}{2}\ \ln \frac{|\mathbf{\Lambda} + \Delta\mathbf{\Lambda}|}{|\mathbf{\Lambda}|}. \tag{7}$$

Taking the natural logarithm, this measure is expressed in nats.

A straightforward evaluation of global entropy reduction as in (7) is computationally expensive since it requires the computation of the determinants of large matrices. To avoid this problem, we introduce an algebraic manipulation of (7) that allows exact, constant-time computation of the mutual information gain. Expanding (7), using (5), and applying Cholesky decomposition to the inverse of the measurement covariance $\mathbf{\Sigma}_y^{-1} = \mathbf{U}^\top\ \mathbf{U}$, we obtain

$$\mathcal{I} = \frac{1}{2}\ \ln \frac{|\mathbf{\Lambda} + (\mathbf{U}\ \mathbf{H})^\top\ (\mathbf{U}\ \mathbf{H})|}{|\mathbf{\Lambda}|}$$

and using the matrix determinant lemma [35]

$$\begin{aligned}
\mathcal{I} &= \frac{1}{2}\ \ln \frac{|\mathbf{I} + (\mathbf{U}\ \mathbf{H})\ \mathbf{\Lambda}^{-1}\ (\mathbf{U}\ \mathbf{H})^\top|\ |\mathbf{\Lambda}|}{|\mathbf{\Lambda}|} \\
&= \frac{1}{2}\ \ln |\mathbf{I} + \mathbf{U}\ \mathbf{H}\ \mathbf{\Sigma}\ \mathbf{H}^\top\ \mathbf{U}^\top| \\
&= \frac{1}{2}\ \ln |\mathbf{U}\ (\mathbf{\Sigma}_y + \mathbf{H}\ \mathbf{\Sigma}\ \mathbf{H}^\top)\ \mathbf{U}^\top| \\
&= \frac{1}{2}\ \ln(|\mathbf{U}\ \mathbf{U}^\top|\ |\mathbf{\Sigma}_y + \mathbf{H}\ \mathbf{\Sigma}\ \mathbf{H}^\top|) \\
&= \frac{1}{2}\ \ln(|\mathbf{\Sigma}_y^{-1}|\ |\mathbf{S}|)
\end{aligned}$$

with $\mathbf{S}$ the innovation covariance matrix, which, thanks to the sparsity of the Jacobian $\mathbf{H}$, is

$$\mathbf{S}\ =\ \mathbf{\Sigma}_y + [\mathbf{H}_i\ \mathbf{H}_n]\ \begin{bmatrix} \mathbf{\Sigma}_{ii} & \mathbf{\Sigma}_{in} \\ \mathbf{\Sigma}_{in}^\top & \mathbf{\Sigma}_{nn} \end{bmatrix} [\mathbf{H}_i\ \mathbf{H}_n]^\top.$$

Therefore, the previous manipulation transforms the ratio of determinants of large matrices in (7) into a ratio of determinants of matrices with the size of the observations

$$\mathcal{I} = \frac{1}{2}\ \ln \frac{|\mathbf{S}|}{|\mathbf{\Sigma}_y|}. \tag{8}$$

It is interesting to note that even if after a loop closure, the uncertainty shrinks for all poses in the trajectory, the ratio in (8) summarizes all the changes in a single, compact expression.

Since sensor registration is an expensive process, in practical applications, it is convenient to hypothesize whether a candidate link is informative enough before actually aligning the sensor readings. In this case, (8) is first evaluated using an approximation of the measurement covariance. If the result is above a given threshold g, sensor registration is needed to assert data association. The real sensor covariance is computed during sensor registration and can be used to recompute the gain measure to ultimately decide whether or not to update the state with the new link.

V. Efficient State Recovery

The two steps of the basic Pose SLAM, i.e., state augmentation and update, can be implemented in constant time assuming the state mean is available. Moreover, the strategies proposed in the previous section to detect relevant poses and links rely on an efficient computation of both the distance between poses and the information gain, and these measures require the state mean and the joint marginal between the last pose and any other pose from the history. Although representing the map in information form allows a more compact representation than in covariance form, it does not offer direct access to state mean and marginal covariances. This section shows that the joint marginal covariances of the last pose and any other pose from the trajectory can be recovered in constant time when operating in open loop and in linear time when closing a loop.

A. State Recovery in Open Loop

Suppose a loop closure occurred at time l. After the loop is closed, when the robot moves to a new pose x_n, $n > l$, the mean for the new pose can be computed from the state transition model with

$$\mu_n = f(\mu_{n-1}, \mu_u)$$

and its marginal covariance is linearly propagated with

$$\mathbf{\Sigma}_{nn} = \mathbf{F}_n\ \mathbf{\Sigma}_{n-1\,n-1}\ \mathbf{F}_n^\top + \mathbf{Q}.$$

Note that since these covariances do not change during open loop traverse, they can be computed once and stored until the next loop closure.

The cross correlation between the last robot pose and any previous pose $i < n$ is

$$\mathbf{\Sigma}_{in} = \mathbf{\Sigma}_{i\,n-1}\ \mathbf{F}_n^\top.$$

Unfolding this recursive relation, $\mathbf{\Sigma}_{in}$ can be factorized as

$$\mathbf{\Sigma}_{in} = \mathbf{\Phi}_i\ \mathbf{F}^\top \tag{9}$$

with

$$\mathbf{\Phi}_i = \begin{cases} \mathbf{\Sigma}_{il}, & 1 \le i \le l \\ \mathbf{\Sigma}_{ii}\ (\mathbf{F}_{l+1}^\top \dots \mathbf{F}_i^\top)^{-1}, & l < i < n \end{cases}$$

and with $\mathbf{F}^\top = \mathbf{F}_{l+1}^\top\ , \dots,\ \mathbf{F}_n^\top$ the accumulated Jacobian from the last loop closure to the current time slice.

Observe that $\mathbf{F}^\top$ can be updated in constant time as the robot moves. Moreover, since the term $(\mathbf{F}_{l+1}^\top, \dots, \mathbf{F}_i^\top)^{-1}$ is the inverse of the aggregated Jacobian $\mathbf{F}^\top$ at time i, all the information needed to evaluate $\mathbf{\Phi}_i$ is available at time i and can be computed in constant time as well.

Therefore, in open loop, the update of the exact joint marginals is achieved in constant time with the minor price of bookkeeping the mean estimated trajectory, the block-diagonal entries of the covariance matrix, aggregated in a block column matrix $\mathbf{D}$, with $\mathbf{D}_i = \mathbf{\Sigma}_{ii}$, and the factors $\mathbf{\Phi}_i$ used to compute the cross covariances when necessary. This extra storage does not diminish the advantage of a sparse information form representation since it only scales the memory requirements by a constant factor.

B. State Recovery when Closing a Loop

When integrating a loop closure constraint y_n^i, the update is obtained by adding the following increment to the state covariance

$$\Delta\boldsymbol{\Sigma} = -\boldsymbol{\Sigma}\mathbf{H}^\top\mathbf{S}^{-1}\mathbf{H}\boldsymbol{\Sigma}.$$

Using Cholesky decomposition for the inverse of the Kalman innovation $\mathbf{S}^{-1} = \mathbf{V}^\top\mathbf{V}$, the above expression takes the form

$$\Delta\boldsymbol{\Sigma} = -\mathbf{B}\mathbf{B}^\top$$

with

$$\mathbf{B} = \boldsymbol{\Sigma}\mathbf{H}^\top\mathbf{V}^\top.$$

Considering (4) becomes

$$\mathbf{B} = [\boldsymbol{\Sigma}_{(i)}\ \boldsymbol{\Sigma}_{(n)}] \begin{bmatrix}\mathbf{H}_i^\top \\ \mathbf{H}_n^\top\end{bmatrix} \mathbf{V}^\top$$

with $\boldsymbol{\Sigma}_{(i)}$and $\boldsymbol{\Sigma}_{(n)}$ the ith and last block columns of $\boldsymbol{\Sigma}$, respectively. The number of columns of the block column matrices $\boldsymbol{\Sigma}_{(i)}$, $\boldsymbol{\Sigma}_{(n)}$, and $\mathbf{B}$ is determined by the dimensionality of the underlying pose space.

Note that only two sections of the prior covariance matrix are necessary to compute the covariance update: $\boldsymbol{\Sigma}_{(i)}$ and $\boldsymbol{\Sigma}_{(n)}$. As described in the previous section, the last block column $\boldsymbol{\Sigma}_{(n)}$ can be computed using the stored factors $\boldsymbol{\Phi}$ and $\mathbf{F}$, using (9). The ith block column of $\boldsymbol{\Sigma}$ can be recovered solving the following linear system:

$$\boldsymbol{\Lambda}_n\boldsymbol{\Sigma}_{(i)} = \mathbf{I}_{(i)} \tag{10}$$

where $\mathbf{I}_{(i)}$ is the sparse block column matrix with an identity block only at the position corresponding to pose i. For very sparse information matrices, such as the one in Pose SLAM, these systems can be solved in near-linear time using supernodal sparse Cholesky factorization [36].

Since we are only interested in the block diagonal and the last column of the new covariance matrix, we can avoid the quadratic cost of computing the whole $\Delta\boldsymbol{\Sigma}$ and directly compute the corresponding updates

$$\Delta\boldsymbol{\Sigma}_{jj} = -\mathbf{B}_j\mathbf{B}_j^\top \tag{11}$$

where $\mathbf{B}_j$ is the jth block row of $\mathbf{B}$, and

$$\Delta\boldsymbol{\Sigma}_{jn} = -\mathbf{B}_j\mathbf{B}_n^\top. \tag{12}$$

Finally, the mean can be updated with

$$\Delta\boldsymbol{\mu} = \mathbf{B}\mathbf{V}(y_n^i - h(\mu_i, \mu_n)). \tag{13}$$

Equations (11)–(13) can be evaluated linearly and the whole loop closure state update scales with the cost of (10). By rigorously controlling the number of loop closures using the information gain, we obtain a system where the sparsity of $\boldsymbol{\Lambda}_n$ is enforced, and this guarantees the near linear cost of the state update when closing a loop.

VI. Tree-Based Nearest Neighbor Search for Data Association

When using the mutual information gain criteria, the proposed Pose-SLAM system scarcely closes loops. The cost of state recovery when closing a loop is amortized over the periods, where the robot operates in open loop and when the state is augmented in constant time. These significant savings in computational cost shift the bottleneck of information-based Pose SLAM to data association, that is, the search over all previously visited poses to determine good candidates for sensor registration. This search is typically implemented as a linear scan over all poses. The solution we propose here is to organize the information about the entire robot trajectory into a binary tree, exploiting the particular properties of the Pose-SLAM problem. The solution described in this section is based on the factorization introduced in Section V-A to compute the cross covariances in constant time when operating in open loop. Using exact cross covariances in a tree-based search for nearest neighbors has the advantage of avoiding false positives which, in turn, translates to fewer queries for sensor matching.

In our implementation, a leaf in the tree will store the mean, the covariance, and the cross-correlation factor associated with a particular pose (μ_i, $\boldsymbol{\Sigma}_{ii}$, $\boldsymbol{\Phi}_i$). The internal nodes of the tree have to somehow summarize the information of all leaves below them, such that a single test at the internal node allows the discarding/acceptance of a large set of poses as neighbors, speeding up the search. We choose to encode the internal nodes using intervals bounding the pose information for all leaves under the corresponding node.

A. Bounding Pose Similarity Using Interval Arithmetic

Interval arithmetic [37] is an extension of real arithmetic where operations are defined over intervals. For instance, for a couple of intervals $\overline{a} = [\underline{a}, \overline{a}]$ and $\overline{b} = [\underline{b}, \overline{b}]$, we have that $\overline{a} + \overline{b} = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$. If $a \in \overline{a}$ and $b \in \overline{b}$, then interval arithmetic guarantees that $a + b \in \overline{a} + \overline{b}$. Using a similar procedure, bounds are defined for all basic operators and for generic functions.

We store in the internal tree nodes the hulls $\overline{\underline{\mu}}_i$, $\overline{\underline{\boldsymbol{\Sigma}}}_{ii}$, and $\overline{\underline{\boldsymbol{\Phi}}}_i$, of the means μ_i, marginal covariances $\boldsymbol{\Sigma}_{ii}$, and factors $\boldsymbol{\Phi}_i$ for all the leaves below the node (see Fig. 2). Using those hulls, we can bound the probability on the displacement with respect to the current pose for all leaves under the node to be inside the given thresholds for each dimension.

Formally, the relative displacement d (see Section IV-A) can be estimated as an interval-based Gaussian with

$$\overline{\underline{\mu}}_d = h(\overline{\underline{\mu}}_i, \mu_n)$$

$$\overline{\underline{\boldsymbol{\Sigma}}}_d = [\overline{\underline{\mathbf{H}}}_i \quad \mathbf{H}_n] \begin{bmatrix}\overline{\underline{\boldsymbol{\Sigma}}}_{ii} & \overline{\underline{\boldsymbol{\Phi}}}_i\,\mathbf{F}^\top \\ \mathbf{F}\,\overline{\underline{\boldsymbol{\Phi}}}_i^\top & \boldsymbol{\Sigma}_{nn}\end{bmatrix} [\overline{\underline{\mathbf{H}}}_i \quad \mathbf{H}_n]^\top$$

where $\overline{\underline{\mathbf{H}}}_i$ is an interval evaluation of the Jacobian of h with respect to its first parameter. Then, an interval for the probability

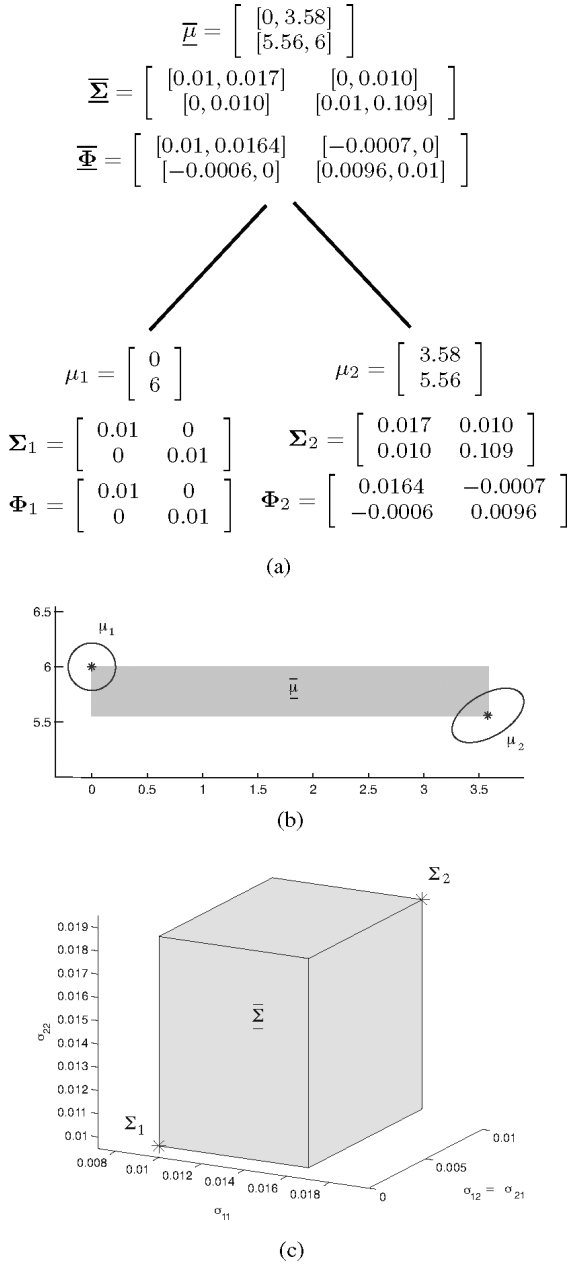

Fig. 2. Example of a tree of poses for the x–y projections of the two first poses of Fig. 1(c). (a) Tree of two poses, showing the pose means, the pose covariances, and the cross correlation factors at each node. (b) The mean hull $\overline{\underline{\mu}}$ for μ_1 and μ_2 is shaded in gray. (c) The covariance hull for the symmetric covariance matrices $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ is shaded in gray.

of the displacement along dimension r being below v_r can be readily computed as

$$\begin{aligned} \overline{\underline{p}}_r &= \int_{-v_r}^{+v_r} \mathcal{N}(\overline{\underline{\mu}}_r, \overline{\underline{\sigma}}_r^2) \\ &= \frac{1}{2}\left(\text{erf}(\overline{\underline{u}}) - \text{erf}(\overline{\underline{l}})\right) \\ &= \frac{1}{2}[\text{erf}(\underline{u}) - \text{erf}(\overline{l}), \text{erf}(\overline{u}) - \text{erf}(\underline{l})] \end{aligned} \tag{14}$$

with

$$\overline{\underline{u}} = \frac{v_r - \overline{\underline{\mu}}_r}{\overline{\underline{\sigma}}_r \sqrt{2}}$$

$$\overline{\underline{l}} = \frac{-v_r - \overline{\underline{\mu}}_r}{\overline{\underline{\sigma}}_r \sqrt{2}}.$$

If $\overline{p}_r < s$, none of the leaves below that node pass the similarity test. On the other hand, if $\underline{p}_r > s$, all the leaves under the node are neighbors of the current pose. For nodes where none of the two previous options hold, the search process has to be recursively applied to the left and right subtrees.

A problem of interval arithmetic is that operations might produce an overestimation of the final result. This happens when we evaluate an expression that includes repeated subexpressions. For instance, the evaluation of $10\,\overline{x} - 8\,\overline{x}$ for $\overline{x} = [1, 5]$ should result in the interval $[2, 10]$, but using simple interval arithmetic, the result is $[-30, 42]$, since the two $\overline{x}$ in the expression are assumed to be independent variables, even though they are not. An excessive overestimation of the bounds of $\overline{\underline{p}}_r$ would vanish the advantage of using a tree. To avoid this risk, the polynomial resulting from (14) is factorized so that each interval variable appears the least number of times possible in the final expression. In the example mentioned earlier, $2\,\overline{x}$ is preferred over $10\,\overline{x} - 8\,\overline{x}$, producing no overestimation.

B. Building a Balanced Tree of Poses

The binary tree of poses is built incrementally adding new poses to a tree as the robot moves and rebalancing the tree when necessary. For the interval evaluation to be accurate, information in the internal tree nodes needs to be as compact as possible; therefore, similar poses need to be grouped under the same node. Since nearby poses along the trajectory are likely to have similar marginal covariance and cross correlation with respect to the current robot pose, we organize the tree such that poses obtained in similar time slices end up in the same branch of the tree. This is achieved by adding new poses always to the same extreme of the tree (the right in our implementation). However, this biased insertion produces unbalanced trees.

Tree balance is obtained in a way similar to what is done with height-balanced binary search trees [38]. The balance factor of a node is the difference between the height of its right and left subtrees. A tree is considered balanced if the balance factor for all its nodes is -1, 0, or 1, where the height of a node is the maximum number of nodes from itself to the leaves. The balance of an unbalanced tree is recovered performing tree rotations. Since insertions only occur at the right-most extreme of the tree, only left rotations are needed to rebalance the tree.

Algorithm 1 shows the pseudocode for the insertion operation, including tree balance. In this Algorithm, LASTNODE($\mathcal{T}$) returns the right-most node of the tree, ADDPOSE($\mathcal{T}, m, i, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i$) adds a new pose to the tree with the provided information as depicted in Fig. 3, ROOT($\mathcal{T}$) returns the root node of the tree, and PARENT(m) returns the parent node of m. Furthermore, BALANCEFACTOR($\mathcal{T}, m$) computes the difference in height between the right and the left subtrees, and LEFTROTATETREE($\mathcal{T}, m$) increases the height of

INSERTTREE($\mathcal{T}, i, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i$)
INPUTS:
$\mathcal{T}$: The tree of poses.
i: The label identifying the new pose.
μ_i: The mean of the pose to add.
$\mathbf{\Sigma}_{ii}$: The marginal covariance of the pose to add.
$\mathbf{\Phi}_i$: The cross covariance factor for the pose to add.
OUTPUTS:
$\mathcal{T}$: The modified tree.

1: $m \leftarrow$ LASTNODE($\mathcal{T}$)
2: $\mathcal{T} \leftarrow$ ADDPOSE($\mathcal{T}, m, i, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i$)
3: **while** $m \neq$ ROOT($\mathcal{T}$) **do**
4: $m \leftarrow$ PARENT(m)
5: **if** BALANCEFACTOR($\mathcal{T}, m$) > 1 **then**
6: $\mathcal{T} \leftarrow$ LEFTROTATETREE($\mathcal{T}, m$)
7: **end if**
8: $\mathcal{T} \leftarrow$ UPDATENODE($\mathcal{T}, m$)
9: **end while**
10: **return** $\mathcal{T}$

Algorithm 1: Insertion of a pose in the tree of poses.

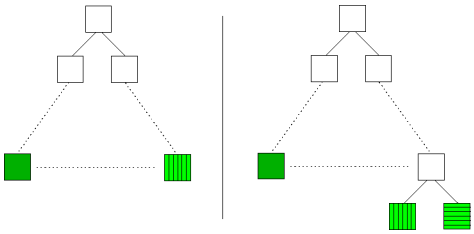

Fig. 3. Adding a new pose to the tree. White boxes represent internal tree nodes and green ones leaves. (Left) Tree before the insertion. (Right) Tree after adding the new pose (marked with horizontal lines). The only pose initially in the tree affected by the insertion is the right-most one, marked with vertical lines.

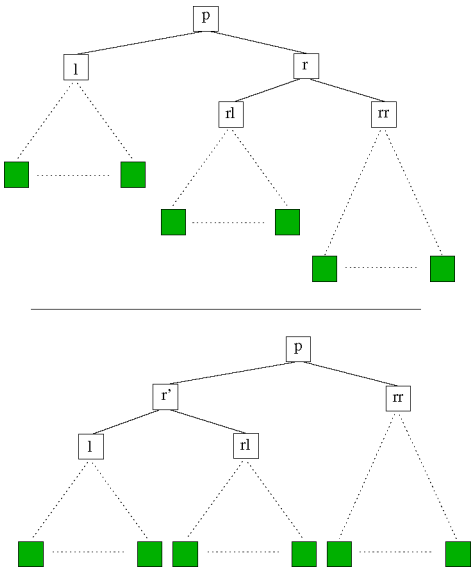

Fig. 4. Rotation of a tree to the left at node P. White boxes represent internal tree nodes and green ones leaves. (Top) Tree unbalanced at node P. (Bottom) Tree after applying the tree rotation.

right child and decreases that of the left one as shown in Fig. 4. Finally, UPDATENODE($\mathcal{T}, m$) redefines the information in an internal tree node as the interval hull of the information stored in the root node of its right and left subtrees.

All operations during insertion have constant time complexity. However, UPDATENODE needs to be applied to the nodes

UPDATETREE($\mathcal{T}, \boldsymbol{\mu}, \mathbf{D}, \mathbf{\Phi}$)
INPUTS:
$\mathcal{T}$: The tree to update.
$\boldsymbol{\mu}$: The state mean.
$\mathbf{D}$: The block diagonal of the covariance matrix.
$\mathbf{\Phi}$: The block last column of the covariance matrix.

1: $m \leftarrow$ ROOT($\mathcal{T}$)
2: **if** HEIGTH($\mathcal{T}$) $= 1$ **then**
3: $i \leftarrow$ LABEL($\mathcal{T}, m$)
4: UPDATELEAF($\mathcal{T}, m, \mu_i, \mathbf{D}_i, \mathbf{\Phi}_i$)
5: **else**
6: UPDATETREE(LEFT($\mathcal{T}, m$), $\boldsymbol{\mu}, \mathbf{D}, \mathbf{\Phi}$)
7: UPDATETREE(RIGHT($\mathcal{T}, m$), $\boldsymbol{\mu}, \mathbf{D}, \mathbf{\Phi}$)
8: UPDATENODE($\mathcal{T}, m$)
9: **end if**

Algorithm 2: Updating the tree of poses.

from the insertion point all the way to the root. Since the tree is balanced, the total cost of inserting a new pose is $O(\log n)$ with n the number of poses already in the tree. Moreover, the memory used by the tree scales with $O(n)$. Finally, note that when a loop is closed the estimates can change considerably. Therefore, the information on all tree nodes must be updated. Updating the tree from scratch is as easy as updating the values stored at the leaves and recomputing the hulls all over the internal nodes. We refer to this procedure as UPDATETREE (see Algorithm 2). In this recursive algorithm, HEIGHT returns the height of a tree (1 for leaves), LABEL returns the label of the pose stored in a given leaf, UPDATELEAVE replaces the data stored in a leaf by the given one, and functions LEFT and RIGHT return the left and right subtrees of a particular node, respectively. The update only visits each node in the tree once; therefore, it has $O(n)$ complexity, which is asymptotically better than the near-linear cost of state recovery at loop closure described in Section V-B.

C. Querying a Tree of Poses

The search (see Algorithm 3) proceeds in a recursive way exploring only those branches that are likely to include poses close to the current one. This is evaluated using the SIMILARITY function that implements (14) for internal tree nodes and (6) for leaves. The information stored in a tree node (interval-based for internal nodes and real for leaves) is obtained using function GETNODEINFO and function LABELS returns the pose labels for all the leaves under a given node. In the end, the search returns the labels associated with the poses that are close to the current one.

In a real mapping situation, we will have an approximately uniform distribution of poses and a reasonable value for threshold s. Thus, each query will return a bounded number of neighbors, say, at most $k \ll n$. Then, the proposed method has an average case complexity of $O(k \log n) = O(\log n)$. In an extreme situation, however, the search process might degenerate. However, in any case, the search process will never visit a node in the binary tree more than once. Thus, asymptotically, the proposed search method is always better or equal, but never worse, than linear search. The degradation toward linear time complexity depends on the distribution of poses and on the neighboring threshold s: If set too low, almost all poses in the trajectory will be selected as neighbors, and the tree-based search degrades to

SEARCHTREE($\mathcal{T}, \mu_n, \boldsymbol{\Sigma}_{nn}, \mathbf{F}, v, s$)
INPUTS:
$\mathcal{T}$: The tree where to search.
μ_n: The mean of the current robot pose.
$\boldsymbol{\Sigma}_{nn}$: The marginal covariance for the current robot pose.
$\mathbf{F}$: Accumulation of Jacobians of the state transition model.
v: Maximum relative displacement from the current robot pose to accept a pose as a neighbor.
s: Minimum probability to accept a pose as a neighbor.
OUTPUTS:
N: The set of labels of the neighboring poses.

1: $m \leftarrow$ROOT($\mathcal{T}$)
2: $(i, \mu_i, \boldsymbol{\Sigma}_{ii}, \boldsymbol{\Phi}_i) \leftarrow$ GETNODEINFO($\mathcal{T}, m$)
3: $\overline{p}_r$ = SIMILARITY($\mu_n, \boldsymbol{\Sigma}_{nn}, \mathbf{F}, \mu_i, \boldsymbol{\Sigma}_{ii}, \boldsymbol{\Phi}_i, v$)
4: **if** $\overline{p}_r \geq s$ **then**
5: $N \leftarrow$ LABELS($\mathcal{T}, m$)
6: **else**
7: **if** $\overline{p}_r < s$ **then**
8: $N \leftarrow \emptyset$
9: **else**
10: $N_l \leftarrow$ SEARCH(LEFT($\mathcal{T}, m$), $\mu_n, \boldsymbol{\Sigma}_{nn}, \mathbf{F}, v, s$)
11: $N_r \leftarrow$ SEARCH(RIGHT($\mathcal{T}, m$), $\mu_n, \boldsymbol{\Sigma}_{nn}, \mathbf{F}, v, s$)
12: $N \leftarrow N_l \cup N_r$
13: **end if**
14: **end if**
15: **return** N

Algorithm 3: Query using the tree of poses.

a linear search. Threshold s should only be low when the robot is almost lost, and in that extreme case, the method would automatically turn into a prior-less data-association mechanism [39].

VII. ALGORITHMIC FRAMEWORK

Algorithm 4 shows the pseudocode for the entire information-based compact Pose-SLAM approach introduced in this paper. To save memory, the algorithm stores the correlation between all poses in information form using $\boldsymbol{\eta}$ and $\boldsymbol{\Lambda}$. Moreover, to efficiently update the state and compute the distance and the mutual information gain, the algorithm keeps track of the state mean $\boldsymbol{\mu}$, the diagonal of the covariance matrix $\mathbf{D}$, and the factors used to recover the last column of the covariance matrix $\boldsymbol{\Phi}$ and $\mathbf{F}$. Finally, the search for neighboring poses is sped up by organizing the information about all poses in the trajectory in a binary tree $\mathcal{T}$. Since the information criteria strictly limits loop creation, the information matrix is sparser than in plain Pose SLAM. The overall algorithm memory footprint is practically linear with the number of poses.

After initializing all data structures (lines 1–9 in Algorithm 4), the algorithm enters the SLAM main loop (lines 10–45). Since redundant poses are removed, at each iteration t, the map includes only n poses, $n \leq t$. In the main loop, we first obtain the new odometric measurement and compound its displacement with respect to the last pose stored in the filter (line 11). This is used to calculate the state mean and relevant parts of the covariance matrix in open loop as described in Section V-A (line 12), as well as to augment the state in information form as described in Section III-A (line 13). Next, the procedure described in Section VI-C is applied to search for neighboring poses (line 14). Applying the method described in Section IV-B, we obtain the most informative link i evaluating, for each link to a possible neighbor, the expected information gain using an approximation to the sensor covariance $\bar{\boldsymbol{\Sigma}}_y$ (line 18). Informative links are queried for sensor matching in decreasing order of information content (line 20). This step is particular for each type of sensor readings to register and returns the constraint between the current and the queried pose parameterized with μ_y and $\boldsymbol{\Sigma}_y$. If sensor registration succeeds (line 21) and the link is actually informative (line 23), the relative measurement is used to recover the state mean and the relevant covariance entries for data association, as described in Section V-B (line 24), as well as to update the state in information form as shown in the formulation in Section III-B (line 25). After closing the loop, the state significantly changes, and the hulls on the tree must be updated (line 26). Subsequent elements in C are then processed, aiming at closing as many loops as possible. Since, at the update, the expected information gain for the remaining candidate loops might significantly change, MAXINFOGAIN must be re-evaluated for the rest of elements in C. After processing all possible loop closures, redundant poses are identified (line 37).

COMPACT POSE SLAM($\mu_0, \boldsymbol{\Sigma}_{00}, v, s, g$)
INPUTS:
μ_0, $\boldsymbol{\Sigma}_{00}$: The mean and covariance of the initial robot pose.
v: Maximum relative displacement from the current robot pose to accept a pose as a neighbor.
s: Minimum probability to accept a pose as neighbor.
g: Minimum information gain to add poses and links.

1: $\boldsymbol{\eta}_0 \leftarrow \boldsymbol{\Lambda}_0 \mu_0$
2: $\boldsymbol{\Lambda}_0 \leftarrow \boldsymbol{\Sigma}_{00}^{-1}$
3: $\mathbf{D}_0 \leftarrow \boldsymbol{\Sigma}_{00}$
4: $\boldsymbol{\Phi}_0 \leftarrow \boldsymbol{\Sigma}_{00}$
5: $\mathbf{F} \leftarrow \mathbf{I}$
6: $\mathcal{T} \leftarrow$ INSERTTREE($\emptyset, \mu_0, \mathbf{D}_0, \boldsymbol{\Phi}_0$)
7: $a \leftarrow 0$
8: $n \leftarrow 1$
9: $t \leftarrow 1$
10: **while** forever **do**
11: $(\mu_u, \boldsymbol{\Sigma}_u) \leftarrow a \oplus$ ODOMETRY
12: $(\mu_n, \mathbf{D}_n, \boldsymbol{\Phi}_n, \mathbf{F}) \leftarrow$ OPENLOOP($\mu_{n-1}, \mathbf{D}_{n-1}, \mathbf{F}, \mu_u, \boldsymbol{\Sigma}_u$)
13: $(\boldsymbol{\eta}_n, \boldsymbol{\Lambda}_n) \leftarrow$ STATEAUGMENT($\boldsymbol{\eta}_{n-1}, \boldsymbol{\Lambda}_{n-1}, \mu_n, \mu_u, \boldsymbol{\Sigma}_u$)
14: $C \leftarrow$SEARCHTREE($\mathcal{T}, \mu_n, \mathbf{D}_n, \mathbf{F}, v, s$)
15: $lowInfo \leftarrow$ FALSE
16: $closed \leftarrow$ FALSE
17: **while** $C \neq \emptyset$ **do**
18: $(i, \mathcal{I}) \leftarrow$ MAXINFOGAIN($C, \bar{\boldsymbol{\Sigma}}_y$)
19: **if** $i < n-1$ **and** $\mathcal{I} > g$ **then**
20: $(\mu_y, \boldsymbol{\Sigma}_y) \leftarrow$ SENSORREGISTRATION(n, i)
21: **if** NOTVOID(μ_y) **then**
22: $\mathcal{I} \leftarrow$ INFOGAIN($i, \boldsymbol{\Sigma}_y$)
23: **if** $\mathcal{I} > g$ **then**
24: $(\boldsymbol{\mu}_n, \mathbf{D}, \boldsymbol{\Phi}) \leftarrow$ CLOSELOOP($\boldsymbol{\Lambda}_n, \boldsymbol{\mu}_n, \mathbf{D}, \boldsymbol{\Phi}, \mu_y, \boldsymbol{\Sigma}_y$)
25: $(\boldsymbol{\eta}_n, \boldsymbol{\Lambda}_n) \leftarrow$ STATEUPDATE($\boldsymbol{\eta}_n, \boldsymbol{\Lambda}_n, \mu_n, \mu_i, \mu_y, \boldsymbol{\Sigma}_y$)
26: $\mathcal{T} \leftarrow$ UPDATETREE($\mathcal{T}, \boldsymbol{\mu}_n, \mathbf{D}, \boldsymbol{\Phi}$)
27: $\mathbf{F} \leftarrow \mathbf{I}$
28: $closed \leftarrow$ TRUE
29: **end if**
30: **end if**
31: **end if**
32: **if** $\mathcal{I} < g$ **then**
33: $lowInfo \leftarrow$ TRUE
34: **end if**
35: $C \leftarrow C \backslash \{i\}$
36: **end while**
37: **if not** $closed$ **and** $lowInfo$ **then**
38: $a \leftarrow (\mu_u, \boldsymbol{\Sigma}_u)$
39: **else**
40: $\mathcal{T} \leftarrow$ INSERTTREE($\mathcal{T}, n, \mu_n, \mathbf{D}_n, \boldsymbol{\Phi}_n$)
41: $a \leftarrow 0$
42: $n \leftarrow n+1$
43: **end if**
44: $t \leftarrow t+1$
45: **end while**

Algorithm 4: Information-based compact Pose SLAM.

TABLE I
MAIN OPERATIONS IN ALGORITHM 4 AND THEIR ASSOCIATED COSTS

Line No.	Operation	Cost
12	OPENLOOP	$O(1)$
13	STATEAUGMENT	$O(1)$
14	SEARCH	$O(\log n)$
18	MAXINFOGAIN	$O(1)$
24	CLOSELOOP	$O(n)$
25	STATEUPDATE	$O(1)$
26	UPDATETREE	$O(n)$
40	INSERTTREE	$O(\log n)$

A pose is considered redundant if it is not used to close any loop and if at least one of the possible links is not informative (line 32). Redundant poses are not added to the filter, but their odometric contribution is simply stored to be used in the next iteration (line 38). Finally, relevant poses are inserted into the search tree (line 40), and the accumulated odometry is reset (line 41).

Table I summarizes the cost of the main steps in the algorithm. In the table, line numbers correspond to those in Algorithm 4. When operating in open loop, the cost of each iteration is dominated by that of the SEARCHTREE. As discussed in Section VI-C, assuming a bounded set of neighbors C, the cost of of the SEARCHTREE is $O(\log n)$, and the cost for computing the MAXINFOGAIN is constant. Loop closure is dominated by the cost of state recovery and by the cost of updating the hulls in the tree, which are both $O(n)$. In conclusion, the cost per iteration in our compact Pose SLAM proposal is well below the current state-of-the-art SLAM algorithms since it is $O(\log n)$ for tree growth, tree search during open loop, and occasionally $O(n)$ for state recovery and tree update at loop closure.

VIII. EXPERIMENTS

This section describes the experiments that validate the presented Pose-SLAM approach, first using synthetic data sets for which ground truth is available to evaluate the quality of the resulting maps and then using several real data sets obtained with our robots and from standard repositories.

A. *Efficiency and Consistency*

The first experiment is designed to show, on one hand, the significant computational savings achieved by restricting the number of poses and links with the presented approach and, on the other hand, that such approach produces a filtering scheme that is less prone to overconfidence from linearization, with the consequent benefit of consistent estimation for longer periods of time.

Fig. 1 shows the reduction in the number of poses and links as a result of using the presented information-based criteria. In the experiment, we simulate a robot moving over two concentric ellipses: the first with semi axes 10 and 6 m and the second with semi axes 20 and 6 m, respectively. In the simulation, the motion of the robot is measured with an odometric sensor whose error is 5% of the displacement in x and y and 0.0175 rad in orientation. A second sensor is able to establish a link between any two poses closer than ± 3 m in x and y, and ± 0.26 rad in orientation. This sensor is simulated with noise covariance $\mathbf{\Sigma}_y = \text{diag}\,(0.2\text{ m}, 0.2\text{ m}, 0.009\text{ rad})^2$.

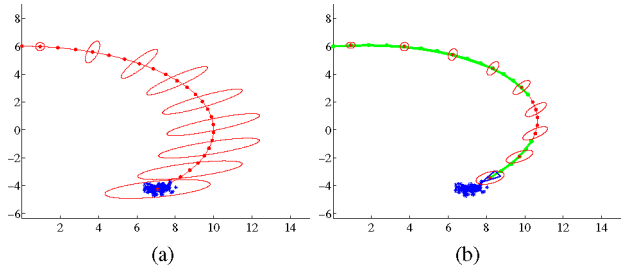

Fig. 5. Monte Carlo simulation of a robot moving on an elliptical trajectory. (a) Open loop. (b) Closing all possible loops.

A map built with the standard Pose-SLAM approach presented in Section III is shown in Fig. 1(a). The result incorporates all possible poses and links to the state. We use the distance test in (6) with $v = (3\text{ m}, 3\text{ m}, 0.26\text{ rad})$ and $s = 0.1$ to determine which poses are to be tested for registration. The simulation takes about 50 s and, at the end of the execution, the filter includes 169 poses and 337 links. All results correspond to a MATLAB implementation running under Linux on a Intel Core 2 at 2.4 GHz. Fig. 1(b) shows results of the same simulation including only links that have a value of (7) higher than 3 nats. In this case, only three links are established in contrast to 337, and the execution time is 28 s.

Finally, Fig. 1(c) shows the outcome of the experiment when only informative links and relevant poses are added to the map. According to Section IV, a pose is considered relevant if a possible link with another pose is informative enough. In this case, the resulting map includes 58 poses and three links, and the simulation takes only 9 s, which is a mere 20% of the time required by the naive implementation from Fig. 1(a).

This gain in speed is not at the cost of map consistency but on the contrary. Fig. 5 shows the evolution of the covariance for the initial 25 steps of a particular run of this experiment. The estimated robot position is indicated by red ellipses. The blue points correspond to 100 Monte Carlo simulations of the robot trajectory using the aforementioned parameters. When keeping all poses but closing only informative loops as in Fig. 1(b), 99% of the sampled trajectories are inside the 95% confidence interval of the last pose covariance, as shown in Fig. 5(a).

Fig. 5(b) shows the linearization effects on the mean and overconfidence on the covariance as a result of adding all possible poses and links, regardless of their information content, as in Fig. 1(a). In this case, at the end only 5% of the Monte Carlo simulated trajectories fall inside the 95% confidence interval of the pose covariance, which exemplifies how a naive filter turns inconsistent sooner when adding many links between poses. This effect is amplified for longer and more complex trajectories [11]. Since in this simulated experiment ground truth is available, we can prove this point by evaluating the normalized estimation error squared (NEES) [40] at each time slice. Fig. 6 shows the NEES at a 95% confidence level when, like in Fig. 1(a), we use all possible links and poses (red dashed line)

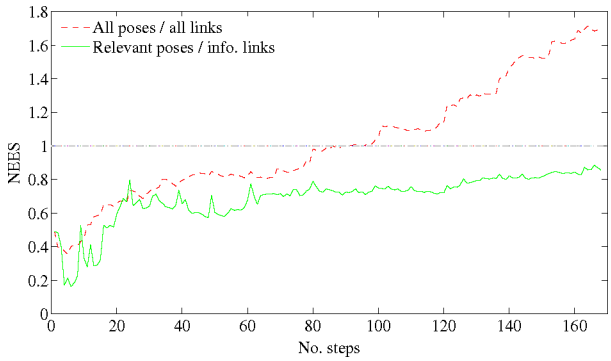

Fig. 6. Consistency evaluation using the NEES criteria.

and the same criteria when, like in Fig. 1(c), we only consider relevant links and poses (green solid line). The lower the NEES, the closer (in the Mahalanobis sense) the estimated trajectory is to the ground truth. NEES values above 1 indicate an inconsistent filter. Clearly, a careful selection of the links and poses to add to the filter resulted in improved filter consistency for the whole mapping session.

B. *Efficient State Recovery*

When closing a loop, the most expensive operation is the partial state recovery described in Section V. We now compare the proposed strategy with two other alternative methods. The first method consists in solving a large sparse linear system to recover the whole covariance matrix $\mathbf{\Sigma}$. The second strategy recovers each block column of the covariance matrix solving separate linear systems. Fig. 7 shows the execution time and memory footprint for these two approaches compared with our method, as a function of state size for the experiment in Fig. 1. Since we are interested in analyzing the price of state recovery at loop closure, the setting in Fig. 1(a) is used, as this is the one in which more loops are closed. In the three experiments, linear systems are solved using the supernodal sparse Cholesky factorization [36], as implemented in the SuiteSparse toolbox [41]. As shown in the plot, the time needed to recover the whole $\mathbf{\Sigma}$ (blue dotted line) is smaller than the time to solve separate system sets: one per each block column (red dashed line). However, memory requirements to solve for the whole $\mathbf{\Sigma}$ increase much faster than when solving the systems column-wise. The method that recovers the whole $\mathbf{\Sigma}$ is according to the memory demanding that it cannot be applied to large mapping problems. Memory use is directly reported by the the SuiteSparse implementation, and the two large jumps in the plot probably correspond to the standard memory-allocation technique that doubles the allocated memory when needed to avoid recurrent expensive system calls. In any case, the execution time and memory usage for our strategy, shown in green solid line in both plots, outperform the two other methods. The experiments with real data in Section VIII-D will show that the computational advantages of the proposed method are also clear for larger data sets.

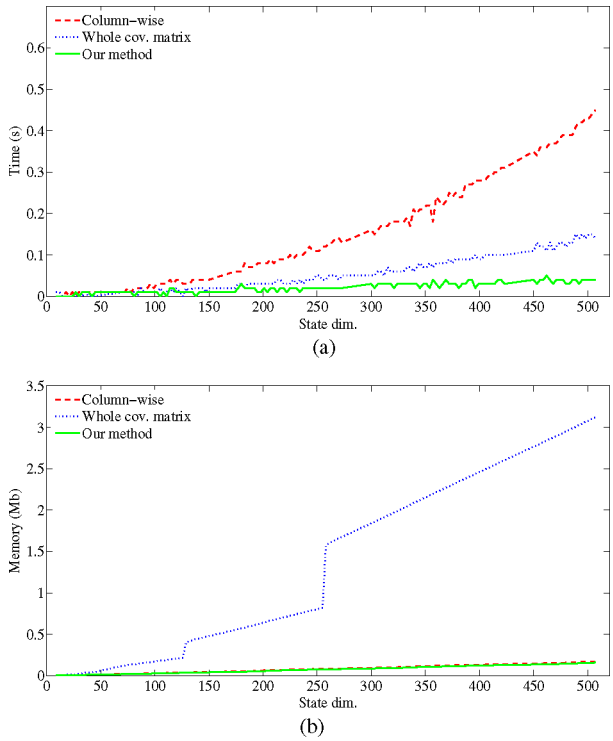

Fig. 7. Execution time and memory footprint for different state recovery strategies when closing a loop for the experiment in Fig. 1(a). (a) Execution time. (b) Memory footprint.

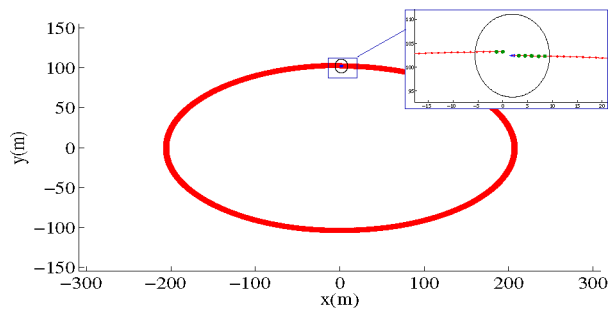

Fig. 8. Example of a simulated trajectory used to compare time execution between linear and tree-based search. The zoom box shows the result of searching for neighboring poses (in green) to the current pose (in blue).

C. *Tree-Based Nearest Neighbor Search*

To compare the proposed tree-based search for neighboring poses with a linear search, we simulate a robot following an elliptical trajectory moving about 1 m per time slice. The size of the ellipse is adjusted so that the range of poses varies from 1000 to 10 000 (see Fig. 8). The robot motion is corrupted with 0.05 m and 0.009 rad of translational and rotational standard deviation, respectively. At the end of the simulation we search for poses within $v = (3\text{ m}, 3\text{ m}, 0.25\text{ rad})$ from the last robot pose, i.e., ± 3 m in x and y and ± 0.25 rad in orientation, with probability higher than $s = 0.5$ in a first run and with a more permissive $s = 0.1$ in a second run.

Fig. 9 shows the execution time in seconds when searching for neighboring poses comparing the linear search versus the tree-based search. Since the trajectories are randomly generated, the

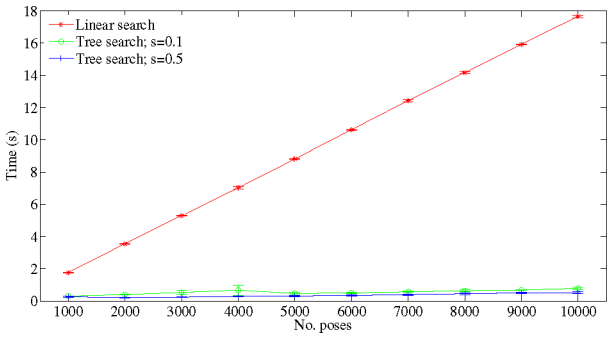

Fig. 9. Averaged execution time in seconds for ten randomly generated experiments using a linear search (red line) and a tree-based search with $s = 0.5$ (blue line) and $s = 0.1$ (green line) for trajectories with 1000 to 10 000 poses. Error bars correspond to one standard deviation.

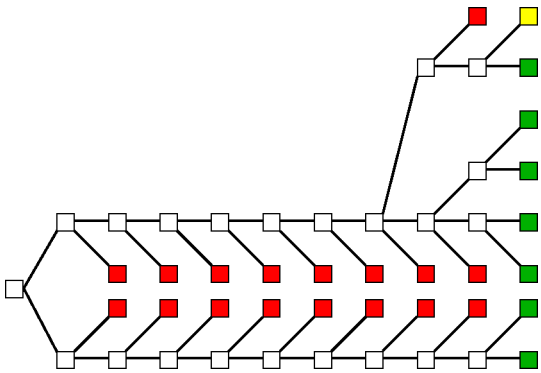

Fig. 10. Portion of the tree explored for a query with 1000 poses. White squares are internal tree nodes that pass the data association test, red squares are tree nodes where this test fails and, consequently, where the recursive search is stopped, yellow squares are leaves where the data association fails, and green squares are the leaves returned as solution for the query.

results are averaged over 10 runs with error bars for one standard deviation. The lower the probability threshold s, the larger the number of poses that pass the neighboring test, potentially reducing the advantage of using the tree-based search. Nonetheless, as shown in Fig. 9, even with $s = 0.1$ the tree-based search hardly degrades, clearly outperforming the linear search, thus validating the approach introduced in this paper.

Fig. 10 shows the portion of the tree explored when solving a query in a tree with 1000 poses for the situation depicted in the inset in Fig. 8. White squares are internal tree nodes that pass the data association test. Red squares are tree nodes where this test fails and, consequently, where the recursive search is stopped. Yellow squares are leaves where the data association fails. Finally, green squares are the leaves returned as solutions for the query. The algorithm returns a set of five poses immediately preceding the current pose in the trajectory and two poses at the beginning of the trajectory. These would be the perfect candidates for loop closure. Since the returned poses are stored at different parts of the tree, two different branches have to be explored to determine the set of results. As shown in the figure, a nice particularity of the interval evaluation is that all branches explored lead to valid match candidates. Thus, the risk of overestimation due to the use of interval arithmetic is avoided in the proposed tree-based data association. Note also how the technique leaves the tree balanced. In this case, for a tree with 1000 poses, the height of the tree is 11.

The need for accurate computation of joint marginals becomes evident during nearest neighbor search. For instance, a standard KD-tree could also be used to speed up the search for neighboring poses. However, in that case, covariances have to be considered in marginal form, bounded at a particular confidence level and approximated by axis-aligned boxes. These approximations would produce many false positives that would later need to be discarded using sensor registration, which is an expensive process. For instance, the number of nearest neighbors identified by our tree-based data-association technique in the whole run of the experiment in Fig. 1 is 362. If, instead, we use a KD-tree that considers covariances in marginal form, the number of nearest neighbors increases to 474, introducing about 30% false positives. In a more-realistic situation, this would imply a considerable increase in the execution time of the algorithm.

D. Real Mapping Sessions

To test the proposed system on real data, we collected dead-reckoning readings and stereo images using a Segway robotic platform fitted with a PointGrey Bumblebee2 stereo rig. Stereo images were used to find constraints (visual odometry and loop closure links) between corresponding poses iterating as follows: First, SIFT image features [42] are extracted and matched from candidate stereo image pairs. The resulting point correspondences are then triangulated to obtain a set of 3-D feature matches, which are in turn used to compute a least squares best-fit pose transformation, rejecting outliers via RANSAC.

The Segway dead-reckoning readings and the visual pose constraints are modeled with noise covariances $\mathbf{\Sigma}_u = \text{diag}\ (0.01\ \text{m}, 0.005\ \text{m}, 0.03\ \text{rad})^2$, and $\mathbf{\Sigma}_y = \text{diag}\ (0.2\ \text{m}, 0.2\ \text{m}, 0.03\ \text{rad})^2$, respectively, and the uncertainty of the initial pose is set to $\mathbf{\Sigma}_{00} = \text{diag}\ (0.1\ \text{m}, 0.1\ \text{m}, 0.09\ \text{rad})^2$. Note that the static motion and measurement covariances are chosen to overestimate the true covariances. This is a worst-case scenario for the proposed nearest neighbor search: The larger the measurement uncertainty, the larger the number of potential neighbors that need to be tested for correspondence. The experiments presented in this paper show that even in this case, the speed up given by the tree-based search is remarkable compared with the linear search. Tighter covariances from more accurate noise models can only result in lower execution times.

Experimentally, we observed that images taken in poses farther away than ± 2.5 m in x, ± 2 m in y, or ± 0.26 rad in orientation cannot be safely matched, and consequently, those are the thresholds in v used to detect nearby poses, with a very permissive $s = 0.1$. With such value of s we avoid missing any valid loop closure.

In the first experiment, we drove the robot for 700 s for about 400 m along two loops around a couple of buildings in the Barcelona Robot Lab, which is part of the EU URUS project, located at the UPC Campus Nord (see Fig. 11). Our Pose-SLAM method was compared with a state-of-the-art place-recognition technique [39]. At a confidence level of 96% for the

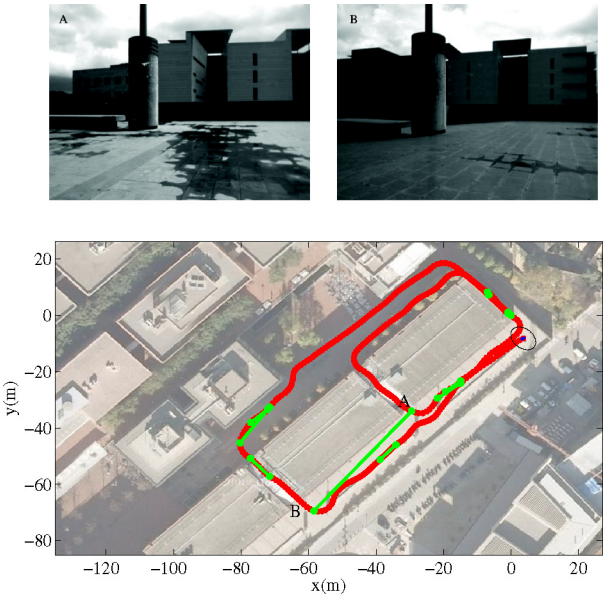

Fig. 11. Poses recognized as being in the same place using a state-of-the-art place-recognition technique at a 95% confidence level linked by green lines.

place-recognition method, only one loop closure could be asserted from the entire data set. Decreasing the confidence level to 95%, all image pairs for the locations linked by green lines in Fig. 11 were recognized as taken from the same locations. The sample images shown (A and B) correspond to images taken when coming out from two different corridors between buildings. Due to its repetitive structure, the environment is strongly aliased; therefore, the use of priors to select candidate links for loop closure is a must.

To see the effect of restricting the number of poses and links with respect to computational time, the same strategies discussed in Section VIII-A are repeated here. When all possible loops are closed [see Fig. 12(a)], we end up with 368 poses and 191 links and an execution time of 182 s, without taking into account vision related processes. When limiting the links to those with information gain above 3 nats, the simulation runs in 131 s, and only three loop closures are established. Finally [see Fig. 12(b)], if we retain only nonredundant poses, we end up with a filter with 147 poses and three loop closures, and an execution time of only 44 s. The computational saving is significant and, as shown in Fig. 12, the final estimate hardly varies with respect to the one using the standard Pose-SLAM approach.

To analyze the effects of using the tree-based nearest neighbor search strategy versus linear search, we report on the case in which all poses are retained, but only informative links are included. Table II gives a comparison of the execution times in seconds. These execution times indicate only filter-related processes (prediction, update, and nearest neighbor search) and do not include sensor-related processes (SIFT extraction and image matching). We can see that in all cases, the time devoted to nearest neighbor search clearly dominates the cost: About 99% of the time is used in this process that clearly motivates the need to improve it. With $s = 0.1$, many poses pass the neighboring test and the advantage of the tree-based search somehow diminishes. Despite this, the use of the tree-based approach reduces the cost of this search by a factor of 0.65. The total time needed to build and rebalance the tree is about 1.5 s only, which is a very small penalty to pay taking into account the computational savings obtained from using it (about 40 s).

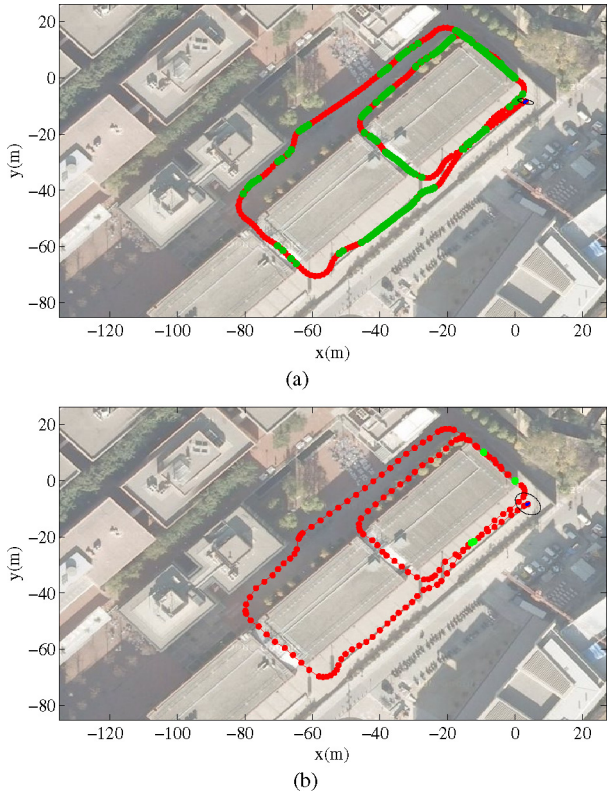

Fig. 12. Filtered trajectory (in red) using encoder and visual odometry on a dataset collected at the first run at the UPC Campus Nord for the standard Pose-SLAM approach and the approach proposed in this paper. Loop closure links are displayed in green, a blue arrow indicates the final pose of the robot, and the black ellipse the associated covariance at a 95% confidence level. (a) Standard approach: incorporating all poses and all links to the filter. (b) Proposed approach: incorporating only relevant poses and links.

TABLE II
EXECUTION TIMES (IN SECONDS) FOR THE FIRST CAMPUS NORD EXPERIMENT

	Search Method	
	Linear-based	**Tree-based**
Nearest Neighbor	131	90
Tree Construction	-	1.5
Loop Closure	0.16	0.16
Open Loop	0.63	0.63

We now report results on the second experiment in which the technique is pushed to stronger conditions on dead-reckoning error with tighter turns and loop-closure assertion with largely overlapped trajectories. In this experiment, we drove the robot for about 1750 s along a more loopy trajectory in an open space in the same Campus. Despite the fact that the trajectory self-intersects many times, the number of loop-closure links is reduced from 5803 to 14 when using the method proposed in this paper. When also removing redundant poses, the number of poses reduces from 981 to 150, and the execution time drops from 1488 to 210 s (almost an order of magnitude faster than

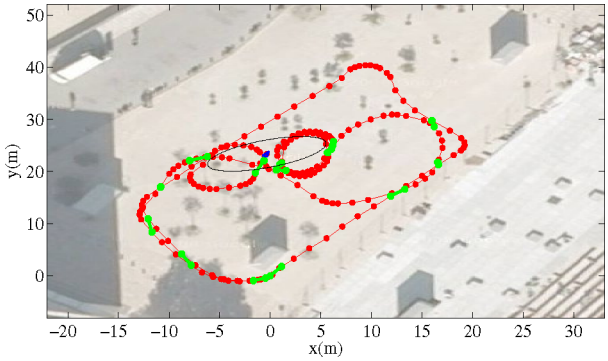

Fig. 13. Filtered trajectory (in red) using encoder and visual odometry on the second dataset collected at the UPC Campus Nord. Loop-closure links are displayed in green, and the blue arrow indicates the final pose of the robot and the black ellipse the associated covariance at a 95% confidence level.

Fig. 14. Filtered trajectory using encoder odometry and laser scans of the Intel dataset. Blue arrow indicates the final pose of the robot and the black ellipse the associated covariance at a 95% confidence level.

real time). These results suggest that, the longer the trajectory, the larger the advantage of the proposed approach. The final estimated trajectory for this experiment is shown in Fig. 13. Note how the density of the poses in the trajectory is automatically adjusted by taking into account the two factors relevant to the information gain: the sensor noise and the uncertainty in the linked poses. For instance, larger noise in rotation automatically accounts for a denser set of poses when the robot rotates as shown in the circular subtrajectory in the center of the figure and is sampled more densely. The consequence is that linking poses using an information criteria produces a trajectory in which poses are equidistant in information space, rather than in Euclidean space. Note that the final marginal covariance for this experiment is larger than that for the previous test. This is due to the fact that the final part of the experiment the robot operates in open loop for a while after closing a mildly informative loop, while in the previous test, an informative loop is closed near the end of the experiment.

Finally, to test the performance for an even larger data set, and to compare the technique with previously published results (see [4] for recent reference), we used the dataset collected at the Intel Research Laboratory (Seattle, WA), which is available at [43]. The dataset includes 26 915 odometry readings and 13 631 laser scans. The laser scans are used to generate sensor-based odometry and to assert loop closures by aligning them using an ICP scan matching algorithm [24]. In this case, only links between poses closer than ± 1 m in x and y, and ± 0.35 rad in orientation were considered reliable. The robot odometry and the relative motion computed from laser scan matches are modeled with noise covariances $\mathbf{\Sigma}_u = \text{diag}\ (0.05\ \text{m},\ 0.05\ \text{m},\ 0.03\ \text{rad})^2$ and $\mathbf{\Sigma}_y = \text{diag}\ (0.05\ \text{m},\ 0.05\ \text{m},\ 0.009\ \text{rad})^2$, respectively. Finally, the covariance of the initial pose is set to $\mathbf{\Sigma}_{00} = \text{diag}\ (0.1\ \text{m},\ 0.1\ \text{m},\ 0.09\ \text{rad})^2$. Using the Pose-SLAM algorithm introduced in this paper with a minimum information gain of 4.5 nats, we end up with a map including only 1218 poses and 103 links. With $s = 0.1$, the total execution time, excluding the ICP scan alignment, is 6314 s. Fig. 14 shows the final estimated trajectory together with the laser scan associated with each of the stored poses in light gray. The blue points in the upper part of the plot correspond to the laser scan for the last robot pose whose covariance in x–y is represented by a black ellipse.

Due to its large size, this dataset is often preprocessed and reduced to about 1000 poses with about 3500 loop closure links. The system we propose automatically selects the optimal subset of poses in the sense of the information gain and not with respect to Euclidean distance, allowing for a more principled selection of loop-closure links and nodes.

Fig. 15 shows the execution time and memory footprint at each step for the different strategies for state recovery at loop closure discussed in this paper: recovering the whole $\mathbf{\Sigma}$, recovering it column-wise, and the method proposed in Section V. The result confirms that for larger SLAM problems, our method outperforms the two other methods both in memory usage and in execution time. Moreover, the results also confirm that the sparsity assumptions behind the efficient loop-closure state recovery, proposed in this paper, actually hold in realistic situations, even when the robot revisits the same area many times. In this particular experiment, the robot travels up to three times around the main corridor of the Intel Laboratory.

In this experiment, the time spent in the two basic filter operations, i.e., state augmentation and state update including the exact computation of state mean and needed covariance terms using the technique introduced in Section V-B, amounts to a very small value of 50 s, i.e., less than 1% of the total execution time, at an average of 4 ms per step. Clearly, this is almost negligible compared with the time needed for data association (6265 s in this case). The advantages of the tree-based search approach proposed in this paper can be better appreciated as the number of poses grows. Larger problems can be formed by processing the data set with lower values for the information gain threshold g. Fig. 16 compares the execution time used to search for nearest neighbors in the last iteration (i.e., when the state includes more poses) of the Intel experiment using the linear

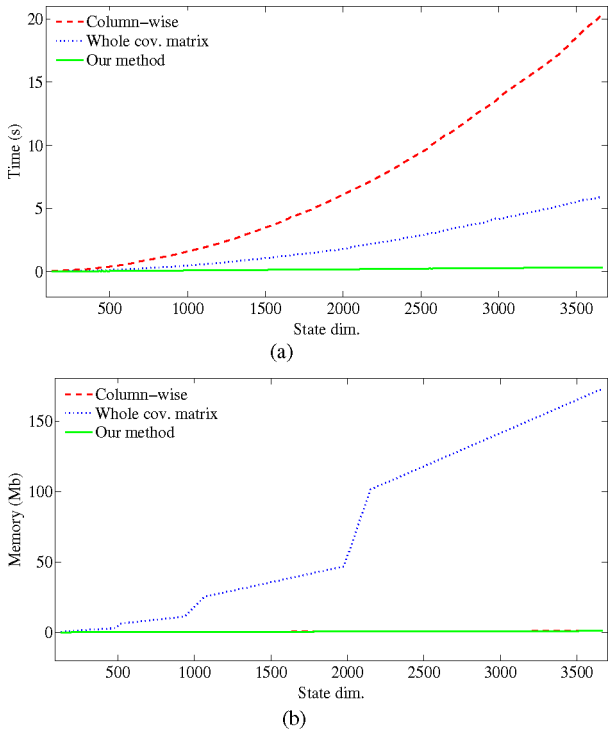

Fig. 15. Execution time and memory footprint for different state-recovery strategies when closing a loop in the Intel experiment. (a) Execution time. (b) Memory footprint.

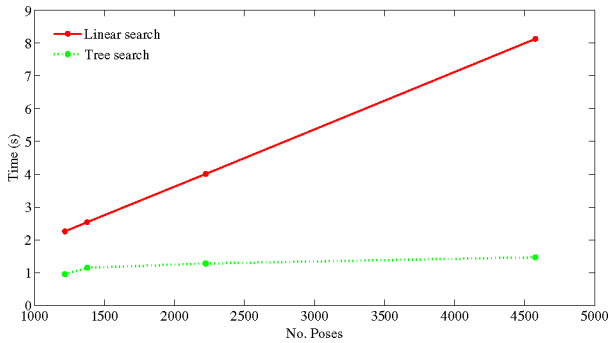

Fig. 16. Time used in the search for nearest neighbors for the last iteration of the Intel experiment processed with different values of the information gain threshold.

search and the tree-based one with $g \in \{4.5, 3.5, 2.5, 1.5\}$. We can see that even when the number of poses substantially grows, the time for the tree-based search hardly increases. These results fully agree with the simulated experiment reported in Fig. 9 and validate the presented search approach in realistic situations.

IX. Conclusion

This paper offers principled solutions to reduce the state size and the number of loop closures in Pose SLAM by considering only nonredundant poses and informative links. This is achieved by computing two measures: the relative distance between poses and the information gain for each candidate link. By storing the state mean and the relevant parts of the covariance matrix, these measures can be computed in exact form and in constant time. Moreover, the paper introduces a constant time procedure to update the mean and the relevant parts of the covariance matrix when operating in open loop and a near-linear time procedure when closing the loop.

With the proposed strategy, the robot operates most of the time in open loop; therefore, the near-linear cost of updating the state after a loop closure is amortized over long periods. In this case, the bottleneck for real-time execution is not state recovery but detecting neighboring poses for which feature matching is likely. We proposed a tree-based method to search for neighboring poses that scales logarithmically with the number of states. Interval arithmetic is used to evaluate the probability of a pose being close to a subset of poses in the trajectory.

Our experiments show that for large mapping sessions the presented technique is beneficial in several fronts: A reduction of state size by an order of magnitude without compromising the quality of estimates is obtained; computationally efficient state recovery is feasible; linearization effects are delayed, maintaining the filter consistent for longer sessions; and neighbor search with the aid of interval arithmetic is able to efficiently identify loop-closure candidates after long periods of open-loop traverse.

Estimation errors have two sources [40]: the approximation introduced by the linearizations and the fact that Jacobians are evaluated at estimates and not at the exact values. In this paper, we addressed the first of this error sources. Maximum-likelihood mapping techniques, e.g., [6], [19], [26], address the second source of error. Possible extensions to our system include the application of the information-based pose and link-selection methods presented in this paper to maximum likelihood techniques in order to improve their efficiency and to obtain a more robust estimator. Another remaining issue is to extend our current implementation to deal with poses in SE(3). Finally, other refinements we would like to address include deriving more elaborated error models and applying the proposed Pose-SLAM algorithm in the context of hierarchical mapping.

References

[1] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *Int. J. Robot. Res.*, vol. 23, no. 7–8, pp. 693–716, Jul. 2004.

[2] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1100–1114, Dec. 2006.

[3] V. Ila, J. Andrade Cetto, R. Valencia, and A. Sanfeliu, "Vision-based loop closing for delayed state robot mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Diego, CA, Nov. 2007, pp. 3892–3897.

[4] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, Dec. 2008.

[5] M. Montemerlo and S. Thrun, *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotic*. (Springer Tracts in Advanced Robotics Series 27). New York: Springer-Verlag, 2007.

[6] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1204, 2006.

[7] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *Int. J. Robot. Res.*, vol. 26, no. 4, pp. 335–359, 2007.

[8] Z. Wang, S. Huang, and G. Dissanayake, "D-SLAM: A decoupled solution to simultaneous localization and mapping," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 187–204, 2007.
[9] K. Konolige and M. Agrawal, "FrameSLAM: From bundle adjustment to realtime visual mapping," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1066–1077, Oct. 2008.
[10] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, Sep. 2003, pp. 1985–1991.
[11] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, May 2001, pp. 4238–4243.
[12] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, China, Oct. 2006, pp. 3562–3568.
[13] K. Konolige, M. Agrawal, and J. Solà, "Large scale visual odometry for rough terrain," presented at the 13th Int. Sym. Robot. Res., Hiroshima, Japan, Nov. 2007.
[14] V. Ila, J. Andrade Cetto, and A. Sanfeliu, "Outdoor delayed-state visually augmented odometry," presented at the 6th IFAC/EURON Symp. Intell. Auton., Vehicles, Toulouse, France, Sep. 2007.
[15] I. Esteban, O. Booij, Z. Zivckovic, and B. Kröse, "SLAM for extremely large environments," presented at the Proc. 14th Annu. Conf. Adv. School Comput. Imag., Heijen, The Netherlands, Jun. 2008.
[16] J. Uhlmann, "Introduction to the algorithmics of data association in multiple-target tracking," in *Handbook of Multisensor Data Fusion*, M. E. Liggins, D. E. Hall, and J. Llinas, Eds. Boca Raton, FL: CRC, 2001.
[17] R. Smith, M. Self, and P. Cheeseman, "A stochastic map for uncertain spatial relationships," presented at the 4th Int. Symp. Robot. Res., Santa Clara, CA, 1988, pp. 467–474.
[18] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, Jun. 2001.
[19] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent," presented at the Robot.: Sci. Syst. III, Atlanta, GA, Jun. 2007.
[20] T. Vidal-Calleja, A. Davison, J. Andrade Cetto, and D. Murray, "Active control for single camera SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, May 2006, pp. 1930–1936.
[21] R. Sim, "Stable exploration for bearings-only SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, Apr. 2005, pp. 2422–2427.
[22] G. Dissanayake, S. B. Williams, H. Durrant-Whyte, and T. Bailey, "Map management for efficient simultaneous localization and mapping (SLAM)," *Auton. Robot.*, vol. 12, no. 3, pp. 267–286, May 2002.
[23] W. Zhou, J. Miro, and G. Dissanayake, "Information-driven 6D SLAM based on ranging vision," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nice, France, Sep. 2008, pp. 2072–2077.
[24] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robot.*, vol. 4, no. 4, pp. 333–349, 1997.
[25] U. Frese, P. Larsson, and T. Duckett, "A multigrid algorithm for simultaneous localization and mapping," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 1–12, Apr. 2005.
[26] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, May 2006, pp. 2262–2269.
[27] A. Ranganathan, M. Kaess, and F. Dellaert, "Loopy SAM," in *Proc. Int. Joint Conf. Artif. Intell.*, Hyderabad, Andhra Pradesh, India, Jan. 2007, pp. 2191–2196.
[28] G. D. Tipaldi, G. Grisetti, and W. Burgard, "Approximated covariance estimation in graphical approaches to SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Diego, CA, Nov. 2007, pp. 3460–3465.
[29] R. M. Eustice, H. Singh, J. J. Leonard, and M. R. Walter, "Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information filters," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1223–1242, 2006.
[30] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 890–897, Dec. 2001.
[31] S. J. Julier and J. K. Uhlmann, "Using covariance intersection for SLAM," *Robot. Auton. Syst.*, vol. 55, no. 1, pp. 2–20, 2007.
[32] S. Huang, Z. Wang, and G. Dissanayake, "Exact state and covariance sub-matrix recovery for submap based sparse EIF SLAM algorithm," in *Proc. IEEE Int. Conf. Robot. Autom.*, Pasadena, CA, Apr. 2008, pp. 1868–1873.
[33] K. L. Ho and P. Newman, "Detecting loop closure with scene sequences," *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 261–286, Sep. 2007.
[34] L. M. Paz, J. D. Tardós, and J. Neira, "Divide and conquer: EKF SLAM in $O(n)$," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1107–1120, Oct. 2008.
[35] D. A. Harville, *Matrix Algebra From a Statistician's Perspective*. New York: Springer-Verlag, 1997.
[36] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM Trans. Math. Softw.*, vol. 35, no. 3, 2008.
[37] R. E. Moore, *Interval Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1966.
[38] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1992.
[39] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, 2008.
[40] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York: Wiley, 2001.
[41] T. Davis. (2009). The SuiteSparse (ver. 3.4). [Online]. Available: http://www.cise.ufl.edu/research/sparse/SuiteSparse
[42] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[43] A. Howard and N. Roy. (2003). The robotics data set repository (Radish). [Online]. Available: http://radish.sourceforge.net

Viorela Ila received the Eng. degree in industrial engineering and automation from the Universitatea Tehnica, Cluj-Napoca, Romania, in 2000 and the Ph.D. degree in information technologies from the Universitat de Girona, Girona, Spain, in 2005.

She is currently a Postdoctoral Fellow with the Institut de Robòtica i Informàtica Industrial, Spanish National Research Council, Barcelona, Spain, as well as a visiting Faculty Member with the College of Computing, Georgia Institute of Technology, Atlanta. Her research interests include state estimation and image processing with applications to mobile and underwater robotics.

Josep M. Porta received the Eng. degree in computer science and the Ph.D. degree in artificial intelligence, both from the Universitat Politènica de Catalunya, Barcelona, Spain, in 1994 and 2001, respectively.

He is currently an Associate Researcher with the Institut de Robòtica i Informàtica Industrial, Spanish National Research Council, Barcelona. His research interest includes planning under uncertainty and computational kinematics.

Juan Andrade-Cetto (S'94–M'95) received the B.S.E.E. degree from CETYS Universidad, Baja California, Mexico, in 1993, the M.S.E.E. degree from Purdue University, West Lafayette, IN, in 1995, and the Ph.D. degree in systems engineering from the Universitat Politécnica de Catalunya, Barcelona, Spain, in 2003.

He is currently an Associate Researcher with the Institut de Robòtica i Informàtica Industrial, Spanish National Research Council, Barcelona. His research interests include state estimation and computer vision with applications to mobile robotics.